1.0 Overview

The Mediation web service provides the capability to search the Department of Defense (DoD) Metadata Registry for Extensible Stylesheet Language Transformation (XSLT) stylesheets. XSLT stylesheets are used to transform an XML document into simple text, HTML, or another XML document. The Mediation web service also provides the capability to perform a transformation on a source document and return the resulting text, HTML, or XML document back to the client application.

1.1 Who should use the web service?

The Mediation web service is available to any client application with a need to locate available XSLT stylesheets and/or perform transformations using those stylesheets.

NOTE: A client application is required to provide a valid username and password in order to perform a search or a transformation. A username and password may be obtained by registering with the DoD Metadata Registry website at: http://xml.dod.mil/mdreg/user/registration/register.cfm.

2.0 Requirements

The Mediation web service satisfies the following requirements:

- The Mediation web service shall provide the capability to search the DoD Metadata Registry for XSLT stylesheets that may be used to transform a particular document.
- The Mediation web service shall provide the capability to search the DoD Metadata Registry for XSLT stylesheets that may be used to transform a document of a particular format or schema.
- The Mediation web service shall provide the capability to search the DoD Metadata Registry for XSLT stylesheets that may be used to produce a document of a particular format or schema.
- The Mediation web service shall provide the capability to apply an XSLT stylesheet to a particular document.

3.0 Detailed Description

The Mediation web service provides several classes and methods. Each is described in the following sections.

3.1 Contents class

The `Contents` class is used to represent the contents of a document such as the input to or output from a transform. The class contains a single data string that is a character representation of a document's contents. The `Contents` class provides the following getter/setter methods.

- `public String getData()`
- `public void setData(String info)`

3.2 TransformInfo class

The `TransformInfo` class contains information about a particular XSLT stylesheet. The following table summarizes the class data.

Field	Type	Description
name	String	Stylesheet name
outputType	String	Type of output produced by the stylesheet (e.g. HTML, XML, …)
sourceFormat	String	Name of the format the source document conforms to.
targetFormat	String	Name of the format the target document conforms to.
namespace	String	Stylesheet namespace
version	String	Stylesheet version
description	String	Stylesheet description

The `TransformInfo` class provides the following getter/setter methods.

- `public String getName()`
- `public void setName(String n)`

- `public String getOutputType()`
- `public void setOutputType(String ot)`

- `public String getSourceFormat()`
- `public void setSourceFormat(String sf)`

- `public String getTargetFormat()`
- `public void setTargetFormat(String tf)`

- `public String getNamespace()`
- `public void setNamespace(String ns)`

- `public String getVersion()`
- `public void setVersion(String ver)`

- `public String getDescription()`
- `public void setDescription(String desc)`

3.3 Transform class

The `Transform` class contains information about a particular XSLT stylesheet as well as the contents of that stylesheet. The class contains two data fields.

Field	Type	Description
info	TransformInfo	Information about the XSLT stylesheet
contents	Contents	Contents of the XSLT stylesheet

The `Transform` class provides the following getter/setter methods.

- `public TransformInfo getTransformInfo()`

- `public void setTransformInfo(TransformInfo ti)`

- `public String getTransformContents()`
- `public void setTransformContents(String data)`

3.5 Mediation_Impl and MediationPort classes

The `Mediation_Impl` and `MediationPort` classes together allow a client application to invoke the Mediation web service. The `Mediation_Impl` class is a stub factory that implements the Java API for XML-based Remote Procedure Calls (JAX-RPC) `Service` interface. The `MediationPort` class is a stub, created via a call to the stub factory.

The constructor of `Mediation_Impl` creates a stub based on the parameter, the location of the Mediation service WSDL file. The `getMediationPort()` method is used to return an instance of the `MediationPort` stub implementation.

The following code shows how to create the `MediationPort` stub:

```
MediationService_Impl ws = new MediationService_Impl(wsdlURL);
MediationServicePort port = ws.getMediationServicePort();
```

Once the stub is created, the client application may invoke the Mediation web service methods. The following code shows a call to the `getAllTransforms()` method:

```
TransformInfo[] transformInfos = port.getAllTransforms(user, password);
```

See the sample code in Appendix A for a complete example client application. The following sections describe the Mediation web service methods.

3.5.1 findSingleStepTransformsFromSource() method

The `findSingleStepTransformsFromSource()` searches the DoD Metadata Registry for XSLT stylesheets that are appropriate for a particular source document. The method requires three parameters.

- The `sourceFormat` parameter is a string whose value is the name of the XML schema to which the source document conforms.
- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns an array of `TransformInfo` objects (see Section 3.2), where each element in the array represents an available XSLT stylesheet that may be used by the client application to transform the source document. If no stylesheets are found, `null` is returned.

```
public TransformInfo[] findSingleStepTransformsFromSource(String
sourceFormat, String user, String password)
throws RemoteException;
```

3.5.2 findSingleStepTransformsFromSourceDoc() method

The `findSingleStepTransformsFromSourceDoc()` searches the DoD Metadata Registry for XSLT stylesheets that are appropriate for a particular source document. The method requires three parameters.

- The `source` parameter is a `Contents` object (see Section 3.1) containing the source document. Because the format (e.g. XML schema name) of the source document is not specified, this method tries to determine the format by parsing the document first. If the caller knows the source document format already, the alternate form of this method (`findSingleStepTransformsFromSource()` – see Section 3.5.1) should be used instead.
- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns an array of `TransformInfo` objects (see Section 3.2), where each element in the array represents an available XSLT stylesheet that may be used by the client application to transform the source document. If no stylesheets are found, `null` is returned.

```
public TransformInfo[] findSingleStepTransformsFromSourceDoc(Contents
source,
String user, String password) throws RemoteException;
```

NOTE: If the format of the source document cannot be determined, the method returns `null`.

3.5.3 findSingleStepTransformsToTarget() method

The `findSingleStepTransformsToTarget()` searches the DoD Metadata Registry for XSLT stylesheets that can produce a result document in a particular format. The method requires three parameters.

- The `targetFormat` parameter is a string whose value is the name of the XML schema to which the result document conforms.
- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns an array of `TransformInfo` objects (see Section 3.2), where each element in the array represents an available XSLT stylesheet that may be used by the client application to produce a target document that conforms to the specified format. If no stylesheets are found, `null` is returned.

```
public TransformInfo[] findSingleStepTransformsToTarget(
String targetFormat, String user, String password)
throws RemoteException;
```

3.5.4 applyTransform() method

The `applyTransform()` method performs the specified transformation on the supplied source document and returns the resulting target document. The method requires four parameters.

- The `source` parameter is a Contents object (see Section 3.1) containing the source document.
- The `transformId` parameter is a string containing the name of the XSLT stylesheet to be used to transform the source document.
- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns a Contents object containing the target document, or `null` if the transform operation fails.

```
public Contents applyTransform(Contents source, String transformId, String
user, String password)
throws RemoteException;
```

3.5.5 getTransform() method

The `getTransform()` method retrieves an XSLT stylesheet from the DoD Metadata Registry database. The method requires three parameters.

- The `transformId` parameter is a string containing the name of the XSLT stylesheet to be retrieved.
- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns a `Transform` object (see Section 3.3) or `null` if the stylesheet was not found.

```
public Transform getTransform(String transformed, String user, String
password) throws RemoteException;
```

3.5.6 getAllTransforms() method

The `getAllTransforms()` method retrieves information about all XSLT stylesheets available in the DoD Metadata Registry database. The method requires two parameters.

- The `user` and `password` parameters are used by the web service to access the DoD Metadata Registry database (see Section 1.1).

The method returns an array of `TransformInfo` objects (see Section 3.2), where each element in the array represents an available XSLT stylesheet.

```
public TransformInfo[] getAllTransforms(String user, String password) throws
RemoteException;
```

4.0 System requirements

- BEA WebLogic Server 8.1 is required for deployment of the web service.
- Java Runtime Environment, version 1.4, is required on the WebLogic server machine, and on client application machines.
- Network connectivity between the WebLogic server machine and the Metadata Registry database server is required.
- Network connectivity between the WebLogic server and client application machines is required.

5.0 Installation / Configuration

The Mediation web service delivery package contains two sets of installation/configuration instructions.

To deploy and configure the service on the WebLogic server, reference:

README-ServerPackage.txt

To install the client library and run the test client, reference:
README-ClientPackage.txt

6.0 Additional Information

For additional information about the Mediation web service, please reference the javadoc files or the installation instructions included with the delivery package.

Appendix A – Sample Client Code

The following code is a sample class called `MediationClient.java`. It provides an example of how a client application might use the Mediation web service to find the XSLT stylesheets appropriate for a specific source document. The location of the Mediation web service WSDL file is required to be the first command line argument. It should be something similar to:

http://diides.ncr.disa.mil:7001/mediation/MediationService?WSDL

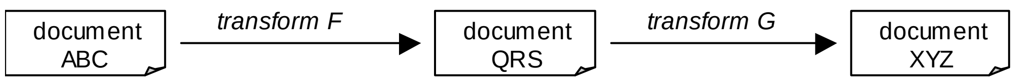
```
public class MediationClient
{
   /**
    * Reads the contents of the specified file into a 'Contents'
    * object.
    * @param inputFile file to read
    * @return 'Contents' object containing file contents
    */
    private static Contents prepareInput(String inputFile)
    {
       Contents contents = null;
       try
       {
          InputStreamReader reader =
        new InputStreamReader(new FileInputStream(inputFile));

          int filesize = 0;
          int c;
          while ((c = reader.read()) != -1)
            filesize++;
          reader.close();

          reader = new InputStreamReader(
                new FileInputStream(inputFile));
          char chars[] = new char[filesize];
          int numCharsRead = reader.read(chars, 0, chars.length);
          contents = new Contents();
          contents.setData(new String(chars));
       }
       catch(Exception e)
       {
          e.printStackTrace();
       }
       return contents;

   }//End prepareInput()


   /**
    * Tests the web service.
    */
   private static void exercise(MediationServicePort service)
                                          throws RemoteException
   {
      String user = "user@fgm.com";
      String password = "user1234";
      try
      {
         //Prepare the source document as a Contents object
         Contents input = prepareInput(sourceDoc);

         //Look for XSLT stylesheets that can transform this source doc
         TransformInfo[] transforms =
              service.findSingleStepTransformsFromSourceDoc(
                                   input, user, password);

         //Print the results, if any found
         if (transforms != null)
         {
            System.out.println("Got " + transforms.length + "
                                        results");
            for (int i = 0; i < transforms.length; i++)
            {
               System.out.println("\tresult: [" +
                             transforms[i].getName() + "]");
            }

         }//end if

      }
      catch(Exception e)
      {
         e.printStackTrace();
      }

   } //end exercise()


   public static void main( String[] args )
   {
      // Setup the global JAXM message factory
      System.setProperty("javax.xml.soap.MessageFactory",
         "weblogic.webservice.core.soap.MessageFactoryImpl");
      // Setup the global JAX-RPC service factory
      System.setProperty( "javax.xml.rpc.ServiceFactory",
          "weblogic.webservice.core.rpc.ServiceFactoryImpl");

      try
      {
         //Use the stub factory to create the stub
         MediationService_Impl ws =
                    new MediationService_Impl(args[0]);
         MediationServicePort port =
                    ws.getMediationServicePort();

         //Run the test.
         exercise(port);
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }

    }//End main()
```

```
} // class MediationClient
```

Appendix B – Known Limitations

Multi-step Transforms

The Mediation web service is only able to search for single-step transforms. In some cases, applying multiple transforms to a source document may be necessary to produce the desired target document. For example, in the picture below, document ABC is transformed into document XYZ via a sequence of two separate transforms.



Currently, the client application has to identify and perform the necessary sequence of transforms to obtain the desired result document. This capability may be provided in a future version of the web service.

Secure Messaging

The SOAP messages passing from a client application to the Mediation web service are not encrypted. Many of the web service methods require `password` parameters that should be secured via encryption. This limitation will be corrected in a future version of the web service.